THE SIMNET VIRTUAL WORLD ARCHITECTURE

James Calvin, Alan Dickens, Bob Gaines,
Paul Metzger, Dale Miller, Dan Owen

LORAL Advanced Distributed Simulation, Inc.
50 Moulton Street
Cambridge, MA 02138

Many tools and techniques have been developed to address specific aspects of interacting in a virtual world. Few, however, have been designed with an architecture that allows large numbers of entities from disparate organizations to interact in such a world, in real time, and over large real geographic distances. This paper describes a system architecture that does this. The paper discusses the key technologies that have made these virtual worlds possible, and explains how the technologies fit into the architecture. A sample implementation of this architecture, the SIMulation NETworking (SIMNET) system, is then presented, along with various design decisions and the reasoning behind them.

1. INTRODUCTION

There are many definitions of "virtual reality" and "virtual worlds." We do not attempt to define these terms here. Instead, we describe an architecture that supports real-time interaction in a virtual world, regardless of how it is defined. This may sound like a lofty claim ("the architecture will support anything you want it to do"), but our years of experience using this architecture show this to be possible. The architecture is flexible enough so that if it does not currently support a task you wish to perform, extending the system to do so requires minimal effort.

In the paragraphs that follow, we describe a system architecture that meets the requirements for operation in a distributed, interactive, virtual world. This architecture is not theoretical. It has been proven and tested over the last decade and has evolved into the largest application of virtual reality in use today (although few people outside key government circles are aware of it). We discuss the technologies that have made this virtual world possible: image generation, virtual-world database modeling, semi-automated forces, networks, and human interface devices. We then describe in detail one of several applications based on this architecture, and discuss various design decisions that were made during its implementation.

2. System Architecture for Virtual Worlds

To begin, we define the requirements of the architecture:

- 100's to 100,000's of entities are supported.
- Entities are geographically distributed.
- Objects (simulations) are heterogeneous.
- System is low cost.
- Computations are distributed (no central site).

0-7803-1363-1/93 $3.00 © 1993 IEEE

- Visual and other spectral domains are supported.
- System operates in real time.

2.1 Key Design Principles

To address the requirements outlined above, we made the following key design decisions that created the foundation for the architecture:

- The model is object- and event-based.

  Objects in the virtual world interact with each other through a series of events. Unchanging objects and information in the virtual world are known by all dynamic objects (i.e., participants in the virtual environment). Dynamic objects keep each other informed of their locations and status through a set of defined events.

- Objects are autonomous.

  All events are broadcast and are available to all interested objects. An object initiating an event does not calculate which other objects might be interested in the event, or how the receiving objects might be affected by it. Each receiving object is responsible for determining which events are of interest to it, and what effect the events may have. One effect of an event may be to generate other events. A side benefit of using autonomous objects is that objects may join or leave a virtual world at any time.

- Objects transmit "ground truth."

  Each object transmits the absolute truth about its state. Receiving objects are responsible for determining whether they can perceive an event and whether they are affected by it. Receiving objects are responsible for transforming ground-truth information to model the real world before presenting this information to a human or to a sensor system. For example, the ground-truth information allows the receiver to know about a vehicle that is hidden behind a near by hill. The receiver is responsible for ensuring that this hidden vehicle is not presented to the receiving vehicle's operator.

- Objects transmit information only about *changes* in their states.

  To minimize requirements imposed on communications processing and bandwidth, objects transmit only *changes* in their behavior. This reduces the transmission (and reception) of redundant information.

- Objects use "dead reckoning" algorithms to extrapolate states.

  Each object extrapolates the new positions of remote objects from the states last reported by those objects. Sending objects are responsible for generating updates before discrepancies become too large. In effect, a "contract" exists between the objects. Each object agrees to maintain a dead reckoning model of itself that corresponds to the model used by remote objects to "see" that object. The updates contain externally "visible" information, including position, velocity, attitude, positions of components (such as a turret and gun barrel on a tank), dust clouds, smoke, muzzle flash, thermal signature, and other electromagnetic emissions. The dead reckoning technique is very efficient compared to sending data at fixed intervals.

In the figure which follows, the position of Object A is shown at times T_n, T_{n1}, and T_{n2}. The example uses dead reckoning threshold discrepancies of 2.5m along all axis. The topmost line labeled "Local position" is the true position of Object A. The line labeled "Remote model position" is the dead reckoned position Ar calculated by Object A with an initial position $A(T_n)$ of (200, 300, 10) and a remote model velocity $Ar\Delta$ of (10, 10, 0). When the difference between

true position A and dead reckoned position Ar exceed the dead reckoning threshold along any axis (as occurs at time T_{n2} in the example), a new packet containing the true position is sent out over the network, and the dead reckoning calculation is restarted.

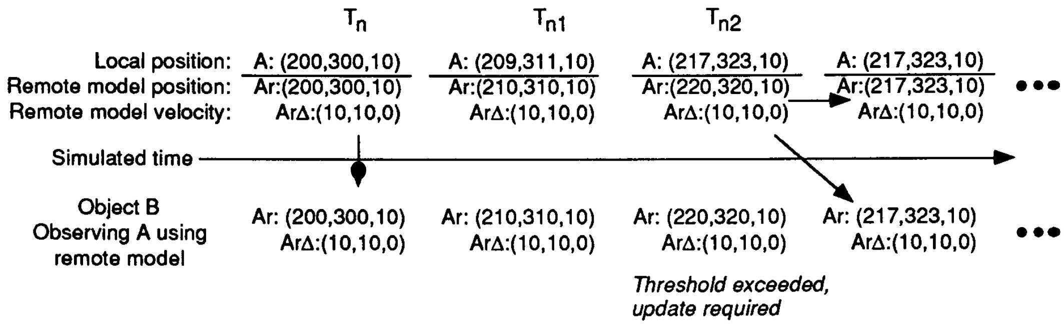

Figure 1: Dead reckoning transmission of state change information

These design decisions allowed us to define an architecture that fulfilled the requirements with the technology available in 1986, but that had the ability to take advantage of new technologies when they became available.

2.2 Enabling Technologies

Having discovered and solidified the key design principles, we then identified the technologies to support the principles and the remaining requirements. We divide these enabling technologies into five major areas:

- Image Generators (IGs)

  IGs provide a visual view into the virtual world. Since virtual worlds are usually presented to human viewers, this visual domain is particularly important. However, the simulation of an object or system also requires feedback about the virtual world in which it exists. Many IGs require the object simulation to maintain a database distinct from the visual database to provide functionality such as terrain feedback, line-of-sight calculations, chord–object intersections, and collision detection.

  It is important that all participants see a consistent view of the virtual world. Some IGs use level-of-detail optimizations that do not work well in this environment. If one object thinks it is hiding in a tree line, an observing object's IG cannot discard the tree line as a level-of-detail optimization, because it would expose the hiding object. Image systems (a superset of an IG) used in a distributed simulation environment must provide a consistent world view, along with the various types of feedback outlined earlier.

- Virtual-world databases

  A virtual world usually exists as a database of some form. In this architecture, the virtual world actually exists in two parts — static and dynamic. The static components are implemented as the IG database and are used for rendering the virtual world. Static components are the passive elements of a virtual environment. The static database provides information that allows objects to detect intersections (such as crashes) with other objects, allows reactions to slope and soil conditions, and may provide line-of-sight support for the simulated object.

  The dynamic components of the virtual world are active participants in the virtual environment; they are the moving or changeable objects in the cyberspace. The database that

describes dynamic objects is distributed across all objects in the virtual world through the change-of-state events. All virtual worlds and the related databases must be derived from a common source to guarantee correlation between databases. Consider what would happen if the terrain and object representations used in one virtual-world representation incorporated, say, a high fidelity representation of mountainous terrain using many polygons, while another representation used lower fidelity and fewer polygons. The participants using the former representation might choose to hide in the craggy terrain, yet the participants using the latter would be able to see them exposed [1].

- Semi-automated forces (SAF)

  Creating virtual worlds with many participants can be difficult. Having enough simulators and people to operate them can be prohibitively expensive. In the military environment that this architecture was designed to support, SAF was created to provide opposing and supporting forces. SAF allows a single human operator to direct the actions of as many as 100 objects in the virtual world. This can be done with directions that are fairly broad, such as *move to a particular geographic location; you figure out the route*, or quite explicit, such as *fire at the specified target*. In addition, SAF objects can react to situations on their own. The important point about SAF objects is that they are indistinguishable from manned objects. This does not imply that SAF objects pass the Turing test, but rather that they use the same underlying principles to interact with other objects in the virtual world. The basic concept of SAF can be applied to almost any domain.

- Networks

  The availability of a high-speed local area network (LAN) was crucial to the development of this architecture. However, the distributed interactive simulation protocol that was developed as part of the architecture is the true enabling concept. This protocol allows the interconnection of many disparate components that might comprise a virtual world. The protocol keeps the virtual world independent of vendor, object domain, and other factors that might limit the applicability of this architecture. Section 3 describes the SIMNET system built using this architecture and some of the SIMNET components. This architecture and the associated protocol are also the basis for the emerging Distributed Interactive Simulation (DIS) protocol [2].

- Human interfaces

  The virtual world supported by this architecture is completely independent of any human interface paradigm. As a result, each object in the virtual world is free to use whatever type of interface is appropriate to that object. In SIMNET, the interface was a fiberglass shell enclosing a set of equipment that replicated the inside of an M1 tank. However, other interfaces have included spatially located sound, a six-degrees-of-freedom control for a "Stealth" vehicle†, and head and eye trackers for various experimental vehicles. The architecture does not require any extensions to support new control and display technologies.

3. SAMPLE IMPLEMENTATION

The SIMNET program was initiated by the Defense Advanced Research Projects Agency (DARPA) in 1983, with substantial support from the U.S. Army. The original intent of the program was to develop the architecture described, and to prove that this architecture was appropriate to support a virtual world that could cost-effectively train soldiers to fight. Since the

†A Stealth vehicle is an observation platform that receives information from the network, but transmits nothing. As a result, it can observe but not be seen.

initial training implementation, SIMNET systems have also been used in combat development applications to test new vehicles and vehicle systems.

From the inception of the SIMNET program, cost concerns were a driving factor in the degree of fidelity that could be provided. Previous simulators had focused on very high-fidelity training, which required a large amount of computational power. They also provided a high degree of realism in the way that controls and displays were presented to the soldiers being trained. Unfortunately, the cost of these simulators was so high that very few could be purchased, and those were aimed at training individual soldiers in highly specialized tasks.

SIMNET simulators, on the other hand, were designed to allow tactical team training involving many soldiers. They were designed to use workstation computers as hosts, and to provide only the controls and displays necessary to train for the tasks specified by the military. As a result, the SIMNET contractors were able to provide a medium-fidelity simulator that was inexpensive enough to be purchased in large quantities suitable for team training. More than 250 SIMNET simulators are currently in operation at sites in the U.S. and Europe.

Figure 2 shows the block diagram of an object using the design principles described in the above sections. Typically, the host processor for such an object is an embedded single-board microprocessor-based system, or a workstation. This simulation host handles the shaded blocks in the diagram.

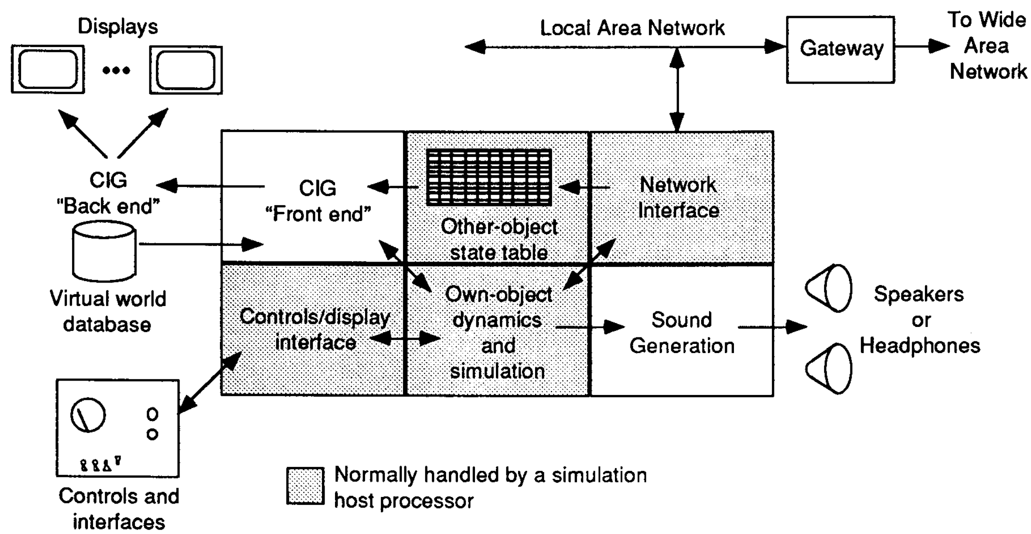

Figure 2: Block diagram of a simulation object

Instances of these objects are interconnected via networks. Usually local area networks (LANs) are used within a single site, and geographically dispersed sites are linked using wide area networks (WANs). Due to the real-time nature of such systems, the WANs are either private tie-lines, or packet networks with gateways that provide real-time allocation capabilities.

One feature that dramatically differentiated SIMNET from previous military simulators was the ability to have many objects playing together in the same virtual battlefield. The key to this is the network communication between objects. During an exercise, each simulator sends messages via the LAN to the other simulators to convey information that they need to know about its appearance and actions. Each simulator also receives, interprets, and responds properly to the

messages received from the other simulators. The success of the SIMNET approach has led to the incorporation of all its essential elements into the DIS draft standard [1]. This standard has recently been accepted by the IEEE as the IEEE 1278 standard for distributed simulations [2].

While none of the interfaces in the simulated vehicles are of especially high fidelity, the completeness of the types of interfaces (visual, aural, tactile, etc.) provides a convincing virtual environment for tactical team training. Indeed, soldiers who train in SIMNET manned simulators sweat, swear, and fight for their virtual lives during engagements. Then, when they are committed to real battles (as in Desert Storm), many attribute their success to the virtual battles fought in SIMNET trainers [3]. SIMNET is indeed, as Howard Rheingold puts it, "video games for real warriors" [4].

4. CONCLUSION

The system architecture being used daily for distributed, interactive simulations has proven to be effective in exploring virtual worlds. Is this system, as it stands today, an off-the-shelf solution for all of the challenges inherent in future virtual realities, worlds, and cyberspaces? Probably not. It is, however, a sound and well-proven baseline. The principles behind the SIMNET and DIS architectures, and the emergence of the IEEE 1278 standard for distributed simulations, provide a wealth of experience, knowledge, and expertise upon which interactive virtual-world applications of all types, military and non-military, can be based.

ACKNOWLEDGMENTS

We would like to acknowledge the following individuals, industry, and government institutions who played key roles in making the development of the SIMNET virtual-world architecture a reality: Col. Jack Thorpe, Col. James Shiflett, the Defense Advanced Research Projects Agency, Perceptronics Inc., the Institute for Defense Analysis, all of the soldiers who helped debug the prototype implementations, and all of the cyberspace pioneers, past and present, from LORAL Advanced Distributed Simulation.

REFERENCES

[1] Miller, Duncan C. The SIMNET architecture for distributed interactive simulation. Presented at the Summer Simulation Conference, Baltimore, MD, July 1991.

[2] Institute of Electrical and Electronics Engineers (IEEE), IEEE P1278 - Standard for Information Technology--Distributed Simulation Application--Process and Data Entity Interchange Formats.

[3] Berry, F. Clifton Jr. Re-creating History - The Battle of 73 Easting. *National Defense*, (November, 1991).

[4] Rheingold, Howard. *Virtual Reality.* Simon & Schuster, New York, 1991, 360-361.

For further information on LORAL Advanced Distributed Simulation products and services, contact Mr. J.C. Williams, Vice President of Business Development, 50 Moulton Street, Cambridge, MA, 02138, or via telephone at (617) 873-3577.